Advanced Software Engineering Engineering Topic

Team orange

Team Members

Team Name : team orange

Members :

s1300016 Tomizawa Kazuto

S1300092 Yuto Kani

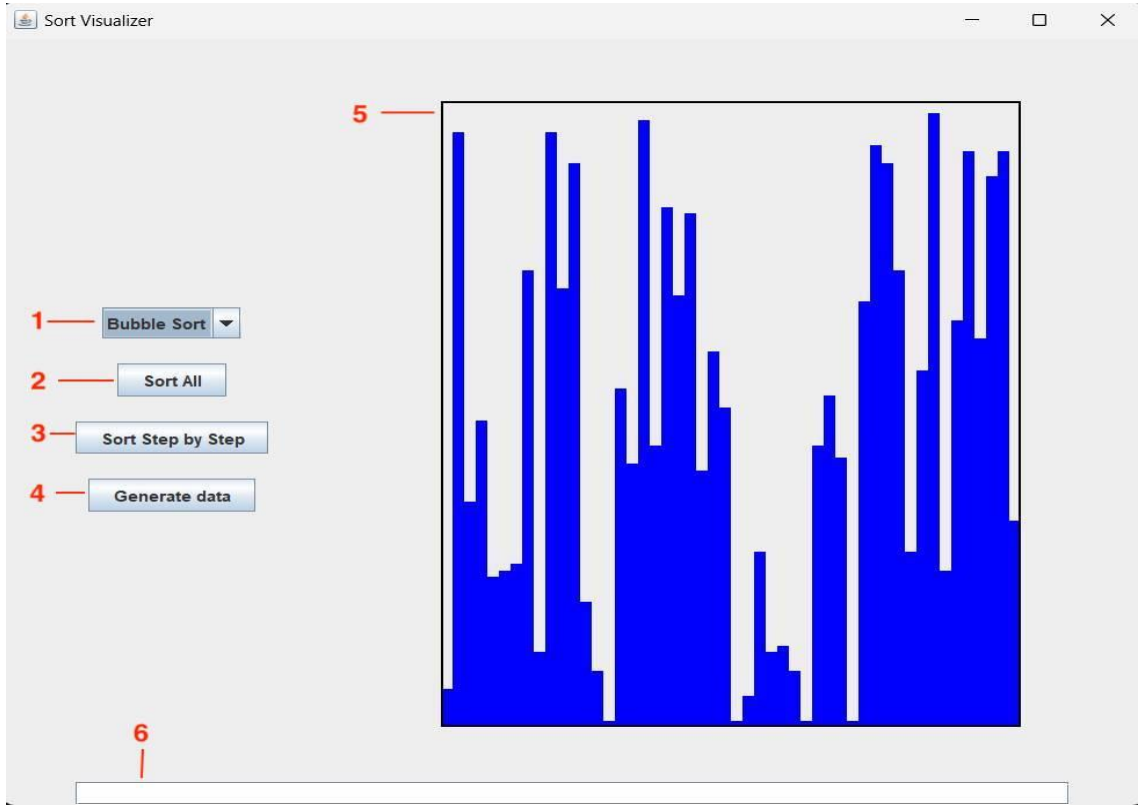
s1300112 Kyoka Obara

Operating environment

- We used JAVA language.
- We used GitHub for software development. (Repository management, branch-based development, task management using issues, etc.)

System Overview

1. Select the sort type
2. Each step is drawn one by one.
3. Only one step is taken and the drawing is updated.
4. Generate new data.
5. The sorting process is displayed according to the buttons pressed.
6. When sorting is complete, the message "Sort complete" will appear. When observing the sort process "Step by step", the number of steps is displayed. For example, "Sort 123/244." (in the case of heapsort)



How to achieve this

Achieving the requirements of the assignment using three types of classes.

- Drawing class
- Sorting class
- Other classes

❖ Drawing class

- Draw the sorting process. Receive the sorting steps from the sorting class in a two-dimensional list, and draw the steps while swapping them accordingly.

MainFrame

❖ Sorting class

- Sorts the given list and returns the order of sorting. For example, it returns the result in the form [[0, 1], [1, 3], [4, 6], [2, 8], ...]. In this case, it swaps 0 and 1, then swaps 1 and 3, and so on.
- Class that inherits ArraylistSorter
- Since it is difficult to return results in this format using merge sort, we create the result in the form of [[0, 15], [1, 23], [4, 16], [2, 38], ...], where the index to be changed and the value after the change are written.

❖ Other class

- Creates random arrays and validates the outcome.

ListUtils

❖ Implementation steps

1. Create a list of random contents with "ListUtils".
2. The sorter sorts the moment you press the button.
3. "MainFrame" is drawn based on the secondary array sent from the sorter that records which items have been swapped.（When you press the "Swap All" button, each step will be drawn one by one. When you press the "Swap Step by Step" button, only one step will proceed, and the drawing will be updated.)
4. When the list of steps is complete, verify it with ListUtils.

Demonstration

Future work

- Some algorithms (MergeSort) require special handling, which causes scalability issues.
- The window size is hard-coded, so when the size of the array to be sorted increases, the display does not function properly.
- Since the change history of the array is managed using a two-dimensional array, there is a possibility of running out of memory if the size of the array to be sorted becomes too large.

Contributions

- Kani : shared ideas on how to proceed, ArraylistSorter interface, MainFrame, QuickSort algorithms;
- Tomizawa : BubbleSort, HeapSort, the ListUtils class, classes for testing sorting algorithms;
- Obara : MergeSort algorithm;

Thank you for listening.